Deep Transfer Learning Ensemble for Classification

Chetak Kandaswamy[1,2,3(✉)], Luís M. Silva[2,4], Luís A. Alexandre[5], and Jorge M. Santos[2,6]

[1] Instituto de Investigação e Inovação em Saúde, Universidade do Porto, Porto, Portugal
[2] INEB - Instituto de Engenharia Biomédica, Porto, Portugal
[3] Department of Electrical and Computer Engineering, Faculty of Engineering, University of Porto, Porto, Portugal
[4] Departamento de Matemática, Universidade de Aveiro, Aveiro, Portugal
[5] Universidade da Beira Interior & Instituto de Telecomunicações, Covilhã, Portugal
[6] Instituto Superior de Engenharia, Politécnico do Porto, Porto, Portugal

Abstract. Transfer learning algorithms typically assume that the training data and the test data come from different distribution. It is better at adapting to learn new tasks and concepts more quickly and accurately by exploiting previously gained knowledge. Deep Transfer Learning (DTL) emerged as a new paradigm in transfer learning in which a deep model offer greater flexibility in extracting high-level features. DTL offers selective layer based transference, and it is problem specific. In this paper, we propose the Ensemble of Deep Transfer Learning (EDTL) methodology to reduce the impact of selective layer based transference and provide optimized framework to work for three major transfer learning cases. Empirical results on character, object and biomedical image recognition tasks achieves that the proposed method indicate statistically significant classification accuracy over the other established transfer learning method.

Keywords: Deep learning · Transfer learning · Ensemble

1 Introduction

Transfer of learning is a well established concept in many fields, including machine learning. Transfer learning (TL) approach in machine learning refers to the procedure employed to train a source model and then transfer the knowledge (learning) across different problems. In principle, it gives better generalization with less computational effort even when the training has a considerable amount of unlabelled data. TL offers several advantages over traditional machine learning specially for non-stationary environments where the training and test samples may be drawn from different marginal distributions or the classification tasks may not be identical.

I. Rojas et al. (Eds.): IWANN 2015, Part I, LNCS 9094, pp. 335–348, 2015.
DOI: 10.1007/978-3-319-19258-1_29

Several viable solutions have appeared in the literature to train machines for non-stationary environments in the past two decades: in lifelong learning [1], where it is assumed that the learner faces an entire family of learning tasks, not just a single one; multi-task learning [2], where it is possible to learn the tasks simultaneously; cross-domain learning [3], where it is possible to reuse the learning when the distributions are correlated; self-taught learning [4], where the machine makes use of large number of easily available unlabelled data to build high-level representations to use for supervised classification tasks.

Recent developments in neural networks inspired by the biological structure of the visual cortex like convolutional networks (CNN) [5], deep belief nets [6], and Stacked denoising autoencoders [7] [8] combined with faster processing computational capabilities lead to the development of deep learning. Deep learning models extract useful information from the input data, constructing multiple levels of representation or learning a hierarchy of features. It potentially leads to progressively more abstract features at higher layers. It is observed that the bottom-layer features are standard regardless of the cost function or dataset used, called as *general,* while the top-layer features depend greatly on the chosen dataset and task, so called as *specific* [9] [11] [12].

The transference of hierarchical features obtained by deep learning for solving classification task, lead to the emergence of Deep Transfer Learning (DTL). DTL an approach in which a deep model is trained on a source problem, and then reused to solve a target problem. In the case of DTL, transference occurs due to two reasons: 1) transferring supervised or unsupervised features from the source problem [9] and 2) retraining only unlocked layers of the network by constraining not to over train for the target task [11].

The DTL approach has proven to be successful in many object and image recognition problems using a layer-by-layer feature transference on large-scale data by transferring hidden layers [10] and retrain unlocked layers [11] [12]. Transferring features of convolutional neural network trained on ILSVRC dataset and retrained to solve several visual classification task on various datasets perform better than state-of-the-art methods [13].

All these above DTL methods have shown that there is a limitation on choosing the various selective layer based transference conditions to solve the new target problem. They do not tackle the problem of negative feature transference and also ambiguous in selecting the layers to be transferred and retrained for the target task.

We may therefore pose the following question:

– Will the learning algorithm be able to adapt by combining the outputs of various selective layer based transference conditions of the deep learning model, in a way, non-negativity constraints as required by the feature transference are no longer needed?

It would be interesting to combine the outputs using ensemble methods. Following the rather standard ensemble methods with deep learning models for classification task. The recent ensemble of deep models using different initial weights on a multi-column CNN in [16], or using combination of different deep

learning methods like CNN and recurrent neural networks in [14], or using combination of shallow and deep models in [15] have shown significant improvement in accuracy. All these ensemble of deep models have shown increased storage and computation.

In this paper, we propose the Ensemble of Deep Transfer Learning (EDTL) approach which combines the advantage of using ensemble of deep models and deep transfer learning. We hypothesise the ensemble of various selective layer based transference on deep models removes non-negativity constraints and speeds up the computation. We study its two conditions of feature transference: 1) transfer specific features, and 2) retrain specific features (splitting of co-adapted neurons). As we will see, EDTL not only effectively reduces the issue of selective layer-based transference as well as improves performance over the established positive transference situations.

We reduce the transferability gap, the gap between the performance of the transference versus the no transference approach increases proportionally to the distance between the source and target distributions [3]. In order to distinguish how different the target distribution is from the source distribution, we use Jensen-Shannon divergence [17] as a metric to measure the degree of heterogeneity between distributions

2 Notations and Problem Settings

Let's represent a dataset by a set of tuples $D = (x_n, y_n) \in X \times Y$, where X is the input space and Y is a set of labels. Assume that the n instances are drawn by a sampling process from the input space X with a certain probability distribution $P(X)$. The dataset is split into subsets of training, validation and test drawn from the same distribution $P(X)$. We assume that the "source" dataset D_S with input space X_S and a set of labels Y_S is drawn from a distribution $P_S(X)$ and the "target" dataset D_T with input space X_T and a set of labels Y_T is drawn from a distribution $P_T(X)$. Such $P_S(X)$ and $P_T(X)$ may be equal or different.

Traditionally, the goal of transfer learning is to transfer the learning (knowledge) from a source problem input space X_S to one or more problems, or distributions to efficiently develop an effective hypothesis for a new task, problem, or distribution [3]. In supervised learning problems, the source and target marginal distribution and classification tasks may be equal or different. In this framework of transfer learning, four possible cases of transfer learning problems can be identified:

We use the well known Jensen-Shannon divergence (JSD) [17] as a measure to compute the difference between two datasets distribution and is given by:

$$D_{JS}(p||q) = \alpha D_{KL}(p||r) + \beta D_{KL}(q||r), \quad \text{with } r = \alpha p + \beta q \tag{1}$$

where D_{KL} is the Kullback-Leibler divergence.

When $\alpha = \beta = 1/2$ in eq.2 we are dealing with the *specific* Jensen-Shannon divergence and D_{JS} is lower- and upper-bounded by 0 and 1, respectively, when using logarithm base 2 [17]. This means that when $D_{JS}(p||q) = 0$ we can consider that p and q are identical and when $D_{JS}(p||q) = 1$, the distributions are different.

Table 1. Transfer Learning cases

Distribution		
Marginal	Labels	Case
$P_S(X) = P_T(X)$	$Y_S = Y_T$	no transfer learning
$P_S(X) \neq P_T(X)$	$Y_S = Y_T$	I
$P_S(X) = P_T(X)$	$Y_S \neq Y_T$	II
$P_S(X) \neq P_T(X)$	$Y_S \neq Y_T$	III

2.1 Established Frameworks

Baseline (BL): Stacked Denoising Autoencoders (SDA) are multiple layer networks where each one is trained as a denoising autoencoder (dA) (see Fig. 1-BL). SDA training comprises of two stages: an unsupervised pre-training stage followed by a supervised fine-tuning stage. During pre-training (PT), the network is generated by stacking multiple dA one on top of each other thus learning *unsupervised features*, represented as a vector $U(w)$ of optimal weights and biases. Then, a logistic regression layer is added on top and the whole network and fine-tuned in a supervised way, thus learning *supervised features* $\mathbf{w} = (w^1, ..., w^K)$, where K is the number of layers.

Transfer Learning (TL): We first train the source network with the source data D_S and Y_S and then copy its hidden layers to the target network. In case $Y_S \neq Y_T$, then we add a classifier layer randomly initialized. The network is trained towards the target task Y_T. If the performance of the newly trained target network exceeds the performance of the baseline approach we have positive transference; otherwise we have negative transference.

Transferred Layers: We select a particular layer or set of layers of the whole baseline network to transfer. For example we may select to transfer first layer features of the baseline approach to the target network, that is, $w_S^1 \Rightarrow w_T^1$. The rest of the target network layer features are randomly initialized.

Retraining Layers: Once the features are transferred to the target network, we add a logistic regression layer for the target task Y_T. We have a choice to fine-tune this entire network $\mathbf{w}_T$ as a multi-layer perceptron using back-propagation or *lock* a layer[11] [12], meaning the transferred feature from source network $\mathbf{w_S^1} \Rightarrow \mathbf{w_T^1}$ do not change during the error propagation for the target task. Thus giving a choice of whether or not to fine-tune the certain layers of the target network. This opens up several possible approaches to solve a problem as shown in Fig .1, TLu and TLs, where the layers are optionally locked or unlocked. This causes fragile co-adaptation of neurons between layers leading to optimization difficulties. The choice of whether or not to fine-tune the first layer of the target network depends on the size of the target dataset and number of parameters [10]. [13]

Transfer Learning Unsupervised (TLu): We transfer the unsupervised features of the SDA model from the source to the target network, i.e., $U(\mathbf{w}_S) \Rightarrow \mathbf{w}_T$ as depicted in Fig. 1, TLu. Once the features are transferred to the target network, we add a logistic regression layer for the target task Y_T. Then we fine-tune the entire classifier like a regular multi-layer perceptron with back-propagation choosing to lock or unlock certain layers to solve the target task.

Transfer Learning Supervised (TLs): The trained weights of the BL approach are used. For example we transfer features from source to target network, i.e., $\mathbf{w}_S \Rightarrow \mathbf{w}_T$. Then we back-propagate choosing to lock or unlock certain layers to solve the target task as illustrated in Fig. 1 TLs.

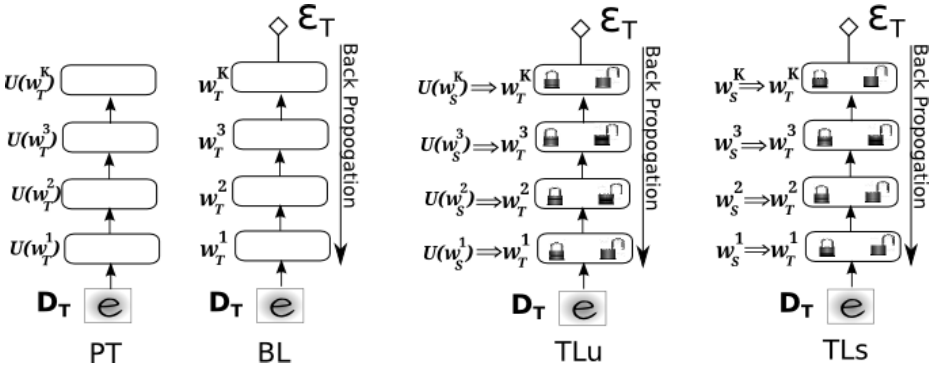

Fig. 1. A pictorial representation of approaches: Pre-training (PT), Baseline (BL), Transfer Learning unsupervised (TLu), and Transfer Learning supervised (TLs).

3 Ensemble of Deep Transfer Learning Framework

In this paper we propose an Ensemble of Deep Transfer Learning (EDTL) where we combine the main advantage of deep transfer learning with the traditional ensemble learning. The DTL offers the knowledge (features) learnt in a source domain providing a good initialization for the learning task in a target problem, better than starting the learning in the target domain at random with possibility of transferring generic features. In here we propose to ensemble the various DTL models between both domains. The intuition is that, like in traditional ensemble, train a DTL model with various transfer and retrain conditions and combine their outputs, treating them as a *committee* of decision makers. Numerous empirical and theoretical studies have demonstrated that ensemble (committee) models often obtain higher accuracy than single models [20].

The overall framework of EDTL depicted in Fig. 2, employs a deep model learnt on the source domain and apply DTL with various conditions, i.e., transfer hidden layer (transfer or randomly initialize) and then retrain (lock or unlock) the network to the target domain. Compute posterior probabilities $P_T(y|x)$ each

Algorithm 1. Pseudocode for baseline and Transfer learning approach

Baseline: Initialize randomly:
Given a two datasets D_A and D_B, Select a dataset and train the network with input x
{**Stage 1:** Pretrain the Network}
build SDA by greedy layer
for k in number of hidden layers **do**
  randomly initialize: W_k
  {Build denoising autoencoder (dA)}
  for each epoch in Pretraining **do**
    Corrupt the input, $x = x + noise$
    $hidden\,layer = Sigmoid(W_k x + bias)$
    $reconstruct = Sigmoid(W'_k x + bias')$
    minimize cross-entropy loss and update weight vector
  end for
  stack the dA's
end for
{**Stage 2:** Fine-tune the Network}
add a logistic regression layer with Y labels
for each epoch in Fine-tuining **do**
  backpropogate the errors
  update the weights
  calculate validation error on validation set
  if best validation error < validation error **then**
    update weights of the network
    best validation error = validation error
    calculate test error on test set
    best test error = current error
  end if
end for
error = best test error

Initialize with trained features D_A:
Given a two datasets D_A and D_B, with tasks Y_A and Y_B,
Select D_A dataset and train the network A as described on the left side.
{**Stage 1:** Transfer the features}
Select a reuse mode: TLu or and TLs
Select which hidden layers to transfer
if $Y_A \neq Y_B$ **then**
  chop of the logistic layer
end if
for k in number of layers **do**
  if layer = transfer **then**
    if mode = TLu **then**
      transfer unsupervised features
      $U(w_A^k) \Rightarrow w_B^k$
    else if mode = TLs **then**
      transfer supervised features
      $w_A^k \Rightarrow w_B^k$
    end if
  else if layer = no transfer **then**
    randomly initialize weights w_B^k
  end if
end for
{**Stage 2:** Fine-tune the Network}
if $Y_A = Y_B$ **then**
  add a logistic regression layer with Y_B labels
end if
for each epoch in Fine-tuining **do**
  backpropogate the errors
  if lock is TRUE in each Layer **then**
    no update of weights
  else
    update the weights
  end if
  calculate validation error on validation set
  if best validation error < validation error **then**
    update weights of the network
    best validation error = validation error
    calculate test error on test set
    best test error = current error
  end if
end for
error = best test error

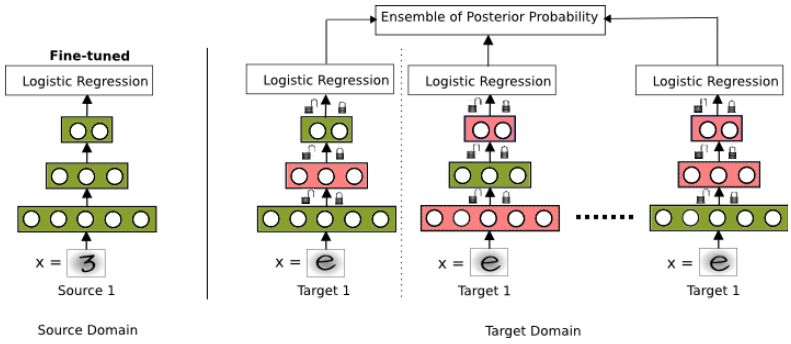

Fig. 2. A pictorial representation of Ensemble of Deep Transfer learning

of the DTL model for target task. Then obtain class-probabilities using ensemble the posterior probabilities $P_T(y|x)$ of each model. The model are trained on baseline method (BL) using standard deep learning approach and the deep transfer learning approach process as listed in Algorithm 1.

The bottom-layer features, called as *general,* similarly the top-layer features, called as *specific*. The pseudo-code for the EDTL process is listed in Algorithm 2, study the two conditions of feature transference: 1) transfer specific features, and 2) retrain specific features (splitting of co-adapted neurons, meaning fragile co-adaptation of neurons is broken by splitting of transferred layer and randomly initialized layer leads to difficulty in optimization [11]).

Algorithm 2. Pseudocode for EDTL

Initialize with trained features D_S**:**
Given two datasets D_S and D_T, with tasks Y_S and Y_T, drawn from P_S and P_T distributions.
Let the total number of models in the ensemble be M
{Select type of TL interaction to evaluate}
if evaluate == co-adapted interactions **then**
    M = possible combination of retrained layers
else if evaluate == generic vs. specific **then**
    M = possible combination of transferred layers
end if
baseline: Train network A using source dataset, D_S as shown in the baseline approach.
for each model M in the ensemble of TL **do**
    transfer: transfer features from network A to new network B as shown in the transfer learning approach
    Compute posterior probabilities $P_T(y|x)$ for target dataset, D_T.
end for
{Combine all the posterior probabilities $P_T(y|x)$ of each model, M}
Compute $y = argmax \sum_{M_i \in M} P_T(y|x)$

4 Pre-processing of Datasets

We test EDTL on three different types of tasks i.e., character, object and biomedical image recognition using four original datasets[1]. To evaluate all possible TL cases as listed in Table 1, we modified the four original datasets into nine different datasets as listed in Table 2.

4.1 Character Recognition Dataset Processing

We evaluate the framework in two different settings for the character recognition task: We use the *MNIST* dataset P_L which has 60,000 training and 10,000 testing instances with labeled hand-written digits from 0 to 9. Additionally, the

[1] We would like to acknowledge researchers making available their datasets, Center for Neural Science, New York University for MNIST; Microsoft Research India for Chars74k; LISA labs, University of Montreal, Canada for BabyAI shapes; and Broad Institute of Harvard and MIT for MCF7-wt breast cancer cells.

Chars74k dataset was modified to obtain *Lowercase* dataset P_{LC} labelled lowercase letters from a-to-z and, the *Uppercase* dataset P_{UC} labelled uppercase letters from A-to-Z and *Digits* P_D dataset contains digits from 0-to-9. Both MNIST and Chars74k datasets used in our experiments have images with 28 x 28 pixels and a sample of each dataset.

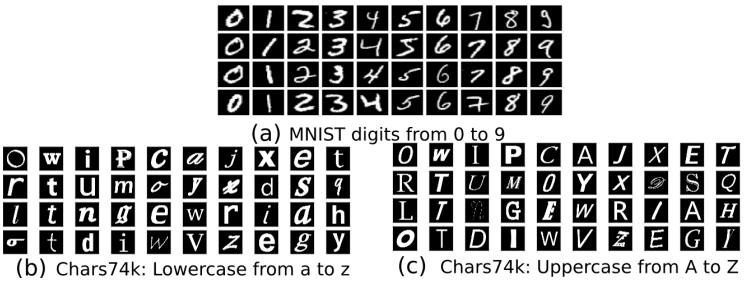

Fig. 3. Samples from character recognition tasks: (a) MNIST, (b) lowercase and, (c) uppercase

4.2 Object Recognition Dataset Processing

We generated three shapes datasets. First, the *canonical* dataset P_{Sh1} has canonical objects, i.e., equilateral triangle, circle and square. Second, the *non-canonical* dataset P_{Sh2} has non-canonical objects, i.e., triangle, ellipse and rectangle. Finally, the *curve Vs. corner* dataset P_{Sh3} has shapes with a curved surface or a corner. All three datasets used in our experiments have images with 28 x 28 pixels and a sample of each dataset.

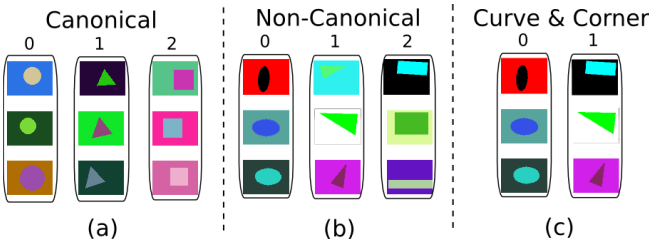

Fig. 4. Samples from various shape recognition tasks: (a) Canonical, (b) Non-Canonical and (c) Curve & corner

4.3 Biomedical Image Recognition Task

We used the BBBC021 image set [18] of genetically engineered MCF7-wt (breast cancer expressing wild-type p53) cell line. The MFC7 dataset has around 4 million cancer cells including Dimethyl sulfoxide (DMSO) control samples. The

CellProfiler software was used to extract 453 features of each of the 148,649 cells of non-control samples. Each single-cell sample was clearly labelled having one of 12 different primary mechanisms of action (MOA) from the subset of compound-concentration combinations [19]. We modified the MFC7 dataset to solve for two different tasks. The first task is to identify 12 different MOA using the single-cell features of non-control samples. The second task is to categorize 38 compounds using the same samples.

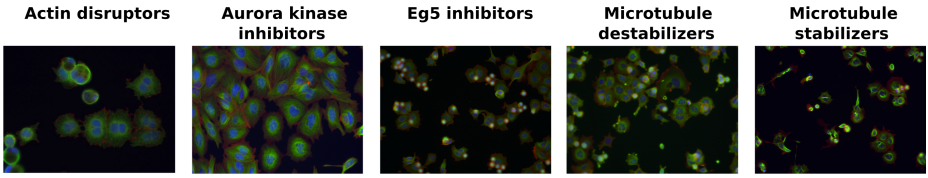

Fig. 5. Examples of the broad spectrum of heterogenic phenotypes captured of MCF7-wt cancer cell assay after compound incubation. The high-content image consists of four-wavelengths, DNA binding dye, DAPI (blue), an actin cytoskeleton marker, Phalloidin (red), tubulin antibody (green) to monitor the microtubule cytoskeleton, and the cytoplasmic marker, HCS cell mask.

Table 2. Number of instances available for each dataset

Data set		**Labels**			**Instances**		
		Ω		classes	**Train**	**Valid**	**Test**
MOA	P_{moa}	0..11	Ω_{12}	12	74,325	37,162	37,162
Compound	P_{com}	0..37	Ω_{38}	38	74,325	37,162	37,162
Lowercase	P_{LC}	a-to-z	Ω_{az}	26	13,208	6,604	6,604
Uppercase	P_{UC}	A-to-Z	Ω_{AZ}	26	13,208	6,604	6,604
Digits	P_D	0-to-9	Ω_{09}	10	13,208	6,604	6,604
MNIST	P_L	0-to-9	Ω_{09}	10	50,000	10,000	10,000
Canonical	P_{Sh1}	eqt,cir,sqr	Ω_{sh1}	3	14,000	1,000	5,000
Non-Canonical	P_{Sh2}	tri,ell,rec	Ω_{sh2}	3	14,000	1,000	5,000
Curve & corner	P_{Sh3}	rou,cor	Ω_{sh3}	2	14,000	1,000	5,000

5 Experimental Setup and Results

Training Deep Neural Network: The network we used in character recognition experiments had three hidden layers with [576, 400, 256] units in order of [bottom, middle, top] respectively, batch size of 100 and pre-training ran for a minimum of 25 epochs. The networks used in object recognition experiments also had three hidden layers with [100, 200, 300] units, batch size of 300 and pre-training ran for a minimum of 10 epochs. The networks used in biomedical image recognition experiments with [453, 906, 1359] units, batch size of 100 and pre-training ran for a minimum of 30 epochs. All the three networks have an output layer appropriate to the number of classes being considered. All hidden layers were pre-trained as denoising autoencoders via gradient descent, using

the cross-entropy cost and a learning rate of 0.001. The complete networks were fine-tuned via gradient descent, using the cross-entropy cost and a learning rate of 0.1. The fine-tuning ran until the validation error did not decrease below 0.1% or until 1000 epochs for all tasks. Our code for experiments was based on the Theano library 6 and ran with the help of an GTX 770 GPU. To determine if a result is statistically significant over ten repetition of each experiment, we used paired student t-test to calculate a p-value, which is the probability of observing an effect given that the null hypothesis is true. We marked each result in Table 3, with '*' when the result was statistically significant, i.e., if an observed p-value is lower than 0.01 (1%).

5.1 Retrain Specific DTL

In this section, we study Retrain specific DTL ($\mathbf{DTL}_r$). In this condition of DTL, we transfer all the hidden layers of the source network to the target network, i.e., transfer [1 1 1 1] and retrain only unlocked layers marked as '1', for example retrain [0 0 1 1]. We study the fragile splitting of the co-adapted neurons caused due to locking of the layer, thus stopping learning in that selected hidden layer of the target network. This avoids overfitting of the network for the target task.

Generally the features of the lower layer of the network are generic therefore they can be used to solve a broader spectrum of problem. The higher layer features are specific to the task the network was trained. We would like to re-utilize the generic features of the source network and retrain the transferred network for target specific task. In this section, we study suitable conditions such that we obtain positive transference retraining only specific layers of the target network.

We observe a consistent improvement in $\mathbf{DTL}_r$ across all the cases of transfer learning for the condition: transfer = [1111] & retrain = [1111]. We conclude that this is due to two main reasons: 1) the transferred layer weights are better than random initialization and, 2) retraining the network target task improves the chances of better generalization.

We observe statistically significant result for all conditions of $\mathbf{DTL}_r$ except for transfer = [1111] & retrain = [0001]. This still offer good generalization than the random accuracy, but is lower than in other conditions.

Ensemble of 4 $\mathbf{DTL}_r$ models gives retrain specific EDTL ($\mathbf{EDTL}_r$). We observe better average accuracy than BL and $\mathbf{DTL}_r$ conditions and results are shown in Table 3. We perform paired student t-test comparing the accuracy results $\mathbf{EDTL}_r$ with accuracy results of $\mathbf{DTL}_r$.

5.2 Transfer Specific DTL

In this section, we study Transfer specific DTL ($\mathbf{DTL}_t$). In this condition of DTL, we transfer only specific layers of the source network to the target network, for example transfer [0 0 1 1] and retrain all the layers, i.e., retrain [1 1 1 1]. We study the generic versus specific feature transference due to transferring of the

Table 3. Percent average classification accuracy obtained for all three possible transfer learning cases; 6 different experiments are performed on three different types of tasks i.e., character, object and biomedical image recognition; We compare established frameworks i.e., Baseline (BL), retrain specific DTL ($\mathbf{DTL}_r$), and transfer specific DTL ($\mathbf{DTL}_t$) with our approach, retrain specific EDTL ($\mathbf{EDTL}_r$), transfer specific EDTL ($\mathbf{EDTL}_t$), and Ensemble of DTL (EDTL); the difference between two datasets distribution and is given by Jensen-Shannon divergence (JSD)

	Marginal	$P_S(X) \neq P_T(X)$		$P_S(X) = P_T(X)$		$P_S(X) \neq P_T(X)$	
	Labels	$Y_S = Y_T$		$Y_S \neq Y_T$		$Y_S \neq Y_T$	
	TL case	I		II		III	
	Experiment	①	②	③	④	⑤	⑥
	Source	Non-Canonical	MNIST	Non-Canonical	COMP	MNIST	MNIST
	Target	Canonical	Digit	Curve & corner	MOA	Lower	Upper
	JSD	0.99	0.99	0	0	0.80	0.79
Approaches		**Avg Acc**	**Avg Acc**	**Avg Acc**	**Avg Acc**	**Avg Acc**	**Avg Acc**
BL		99.49(0.32)	97.74(0.09)	98.35(0.27)	96.38(0.5)	94.34(0.13)	94.93(0.13)
Retrain Specific DTL							
	transfer retrain						
$\mathbf{DTL}_r$	[1111] [1111]	99.51(0.17)	97.92(0.27) *	99.00(0.46)	97.54(0.40) *	94.71(0.22) *	94.94(0.23) *
	[1111] [0111]	96.92(1.72) *	98.06(0.17) *	97.06(1.47) *	98.07(0.18)	94.52(0.24) *	94.74(0.15) *
	[1111] [0011]	96.60(1.64) *	98.14(0.18) *	96.79(1.53) *	98.23(0.17)	93.92(0.29) *	94.10(0.15) *
	[1111] [0001]	95.78(1.91) *	97.36(0.51) *	96.46(1.68) *	98.16(0.22)	89.36(0.83) *	89.57(0.99) *
$\mathbf{EDTL}_r$		**99.57**(0.13)	**98.62**(0.14)	**99.27**(0.31)	**98.24**(0.27)	**95.11**(0.18)	**95.38**(0.18)
Transfer Specific DTL							
	transfer retrain						
$\mathbf{DTL}_t$	[1111] [1111]	99.51(0.17) *	97.93(0.27) *	99.00(0.46) *	97.54(0.40)	94.71(0.22)	94.94(0.23) *
	[0111] [1111]	99.73(0.12) *	97.09(0.34) *	**99.60**(0.16) *	96.76(1.22)	92.80(0.26) *	93.96(0.15) *
	[0011] [1111]	86.83(17.26)	97.29(0.30) *	68.81(5.71)	96.71(0.46) *	93.28(0.28) *	93.97(0.33) *
	[0001] [1111]	99.84(0.08)	97.46(0.17) *	98.86(0.38) *	96.81(0.65)	93.40(0.17) *	94.38(0.16) *
$\mathbf{EDTL}_t$		**99.91**(0.03)	**98.18**(0.16)	99.58(0.16)	**97.54**(0.53)	**94.56**(0.11)	**95.26**(0.10)
EDTL		99.86(0.03)	**98.87**(0.10)	99.52(0.16)	98.12(0.31)	**95.18**(0.06)	**95.70**(0.16)

layer, thus reusing the features for the target task. This not only speeds up the training but also improves the accuracy of the network.

We observe that $\mathbf{DTL}_t$, even for condition when only the logistic regression layer is transferred and retaining the whole target network with backpropagation algorithm had better accuracy than the BL as shown in Table 3.

Ensemble of 4 $\mathbf{DTL}_t$ models gives transfer specific EDTL ($\mathbf{EDTL}_t$). We observe better average accuracy than BL and $\mathbf{DTL}_t$ conditions. Results are shown in Table 3.

5.3 Ensemble of both $\mathbf{EDTL}_r$ and $\mathbf{EDTL}_t$

We observe significant improvements in average accuracy using EDTL over both $\mathbf{EDTL}_r$ and $\mathbf{EDTL}_t$ using all the conditions as listed in Table 3 except for the transfer learning case II. Firstly, we observe that in $\mathbf{EDTL}_r$, 6 out of 6 experiments obtains better accuracy than BL and other established DTL approaches. Secondly, we observe that in $\mathbf{EDTL}_t$, 5 out of 6 experiments obtains better accuracy than BL and other established DTL approaches. Finally, we observe in EDTL, 3 out of 6 experiments obtains better accuracy than BL and other established DTL approaches.

6 Conclusions and Discussion

We propose an ensemble of transfer learning approaches using 9 datasets with varied image recognition tasks like character, object and biomedical image recognition. We make several contributions as listed below:

1. We analyse all possible cases of transfer learning, i.e., based on change in distribution and based on change in classification task between the source and the target domains.

2. The experimental analysis of the retrain specific DTL approaches across all possible cases of transfer learning show that the condition: transfer all layers and retain all layers, obtains better overall accuracy not only than baseline but also compared to other DTL_r conditions. This is due to two main reasons: 1) the transferred layer weights are better than random initialization and, 2) retraining the network target task improves the chances of better generalization, i.e., by forcing splitting of fragile co-adapted neurons the network avoids overfitting to the target task.

3. We observe that transfer specific DTL approaches obtain better overall accuracy than baseline but not as good as retrain specific DTL, as fine-tuning of randomly initialized weights forces the solution to local minima.

4. The experimental analysis of retrain specific EDTL, transfer specific EDTL and EDTL approaches show that $EDTL_r$, ensemble of posterior probabilities of four DTL_r models, obtain a statistically significant better accuracy than individual DTL_r. EDTL outperforms baseline and other DTL approaches both when the distributions and task are different.

In future we would like to explore the possibility of training multiple source domain problems and combine them under ensemble of deep transfer learning framework.

Acknowledgments. This work was financed by FEDER funds through the *Programa Operacional Factores de Competitividade* COMPETE and by Portuguese funds through FCT Fundação para a Ciência e a Tecnologia in the framework of the project PTDC/EIA-EIA/119004/2010.

References

1. Thrun, S.: Learning to learn: Introduction. In Learning To Learn (1996)
2. Caruana, R.: Multitask learning. Machine Learning **28**(1), 41–75 (1997)
3. Daumé III, H., Marcu, D.: Domain Adaptation for Statistical Classifiers. J. Artif. Intell. Res. (JAIR) **26**, 101–126 (2006)
4. Raina, R., Battle, A., Lee, H., Packer, B., Ng, A.Y.: Self-taught learning: transfer learning from unlabeled data. In: Proc. of the ACM Conference on (ICML), pp. 759–766 (2007)
5. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. proceedings of the IEEE **86**(11), 2278–2324 (1998)
6. Hinton, G.E., Osindero, S., Teh, Y.: A fast learning algorithm for deep belief nets. The Journal of Neural computation **7**, 1527–1554 (2006)
7. Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.-A.: Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. J. Mach. Learn. Res. **11**, 3371–3408 (2010)
8. Bengio, Y., et al.: Towards Biologically Plausible Deep Learning. arXiv preprint arXiv:1502.04156 (2015)
9. Kandaswamy, C., Silva, L., Alexandre, L., Sousa, R., Santos, J.M., Marques de Sá, J.: Improving transfer learning accuracy by reusing Stacked Denoising Autoencoders. In: IEEE Conference on Systems Man and Cybernetics. IEEE (2014)
10. Kandaswamy, C., Silva, L.M., Alexandre, L.A., Santos, J.M., de Sá, J.M.: Improving deep neural network performance by reusing features trained with transductive transference. In: Wermter, S., Weber, C., Duch, W., Honkela, T., Koprinkova-Hristova, P., Magg, S., Palm, G., Villa, A.E.P. (eds.) ICANN 2014. LNCS, vol. 8681, pp. 265–272. Springer, Heidelberg (2014)
11. Yosinski, J., Clune, J., Bengio, Y., Lipson, H.: How transferable are features in deep neural networks? In: Advances in Neural Information Processing Systems, pp. 3320–3328 (2014)
12. Kandaswamy, C., Silva, L., Cardoso, J.S.: Source-target-source classification using Stacked Denoising Autoencoders. In: Proc. of the 7th Iberian Conference on Pattern Recognition and Image Analysis, Santiago de Compostela, Spain, June 2015
13. Razavian, A.S., Azizpour, H., Sullivan, J., Carlsson, S.: CNN features off-the-shelf: an astounding baseline for recognition. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 512–519. IEEE (2014)
14. Deng, L., Platt, J.C.: Ensemble deep learning for speech recognition. In: Proceedings of the Annual Conference of International Speech Communication Association (INTERSPEECH) (2014)
15. Abdullah, A., Veltkamp, R.C., Wiering, M.A.: An ensemble of deep support vector machines for image categorization. In: International Conference of Soft Computing and Pattern Recognition, SOCPAR 2009, pp. 301–306. IEEE (2009)

16. Ciresan, D., Meier, U., Schmidhuber, J.: Multi-column deep neural networks for image classification. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE (2012)
17. Lin, J.: Divergence measures based on the Shannon entropy. IEEE Transactions on Information Theory **37**, 145–151 (1991)
18. Ljosa, V.: Katherine L. Sokolnicki, and Anne E. Carpenter.: Annotated high-throughput microscopy image sets for validation. Nat Methods **9**(7), 637 (2012)
19. Ljosa, V., et al.: Comparison of methods for image-based profiling of cellular morphological responses to small-molecule treatment. Journal of biomolecular screening (2013)
20. Kuncheva, L.I.: Combining Pattern Classifiers: Methods and Algorithms. Wiley Press (2004)